209700-1-T

Technical Report

PERFORMANCE ANALYSIS OF THE DAP-610 AS A DATA COMPRESSION PROCESSOR IN A REAL-TIME REMIDS SYSTEM

L.M. PRZEBIENDA

MARCH 1989

Report for Period November 1988-February 1989

Performed under Contract DAAK70-88-C-0032 funded by
U.S. Army Belvoir Research & Development Center

Submitted to:
U.S. Army Belvoir Research & Development Center
Countermine Technology Division
Detection Research Team STREB-NMD Bldg. #339
Fort Belvoir, Virginia 22060

Attn: Mr. R. Dupont

ERIM P.O. Box 8618 Ann Arbor, MI 48107-8618

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA206587

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited Distribution
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBERS(S) 209700-1-T	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION ERIM 6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION U.S. Army Belvoir Research & Dev. Center Countermine Technology Division
6c. ADDRESS (City, State, and ZIP Code) P.O. Box 8618 Ann Arbor, MI 48107	7b. ADDRESS (City, State, and ZIP Code) Fort Belvoir, VA 22060
8a. NAME OF FUNDING /SPONSORING ORGANIZATION U.S.Army Belvoir Research & Development Ctr. 8b. OFFICE SYMBOL *(if applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAK70-88-C-0032
8c. ADDRESS (City, State, and ZIP Code) Fort Belvoir, VA 22060	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
Performance Analysis of the DAP-610 as a Data Compression Processor in a Real-Time System

12. PERSONAL AUTHOR(S)
L. M. Przebienda

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM 11/88 TO 3/89	1989 March 31	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

An analysis of the DAP™ 610 (Active Memory Technology's Distributed Array Processor) was undertaken to determine if the DAP-610 is capable of functioning as a major system component in a real-time processor. The evaluated functions required of the DAP-610 include: a direct link to an external multispectral sensor sending 16-bit data at a 1.65MHz rate, demultiplexing and reformatting of sensor data, data compression processing, feature extraction, VMEbus ~~I/O~~ for communication with other system components, and output of high resolution image data for display. The DAP-610 was found to be capable of all these functions. However, processing time estimates available are insufficient to verify that the implementation will operate within the real time constraints.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ OTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR

UNCLASSIFIED

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

At the request of the Waterways Experiment Station (WES), the Environmental Research Institute of Michigan evaluated the performance of the DAP^{TM} 610 (Active Memory Technology's Distributed Array Processor) hardware for data compression in the Real-Time Processor (RTP). All functions expected of the DAP 610 as the Compression Processor are presented in this report.

Figure 1 illustrates the functions expected of the DAP 610 as it is configured in the Real-Time Processor. The DAP is shown directly linked to an external sensor that will provide multispectral images to the DAP. The DAP must properly format the incoming image data and then in real time apply the data compression algorithm. Data compression results will be output through the VMEbus to the Classification Processor. Classification results will be returned to the DAP through the VMEbus and overlaid on one of the original three images buffered by the DAP. The DAP must vertically compress the buffered image and then output the image from the frame buffer board in either a page or scroll mode for image display. Output of an image for display will be synchronized with completion of operations in the Classification Processor.

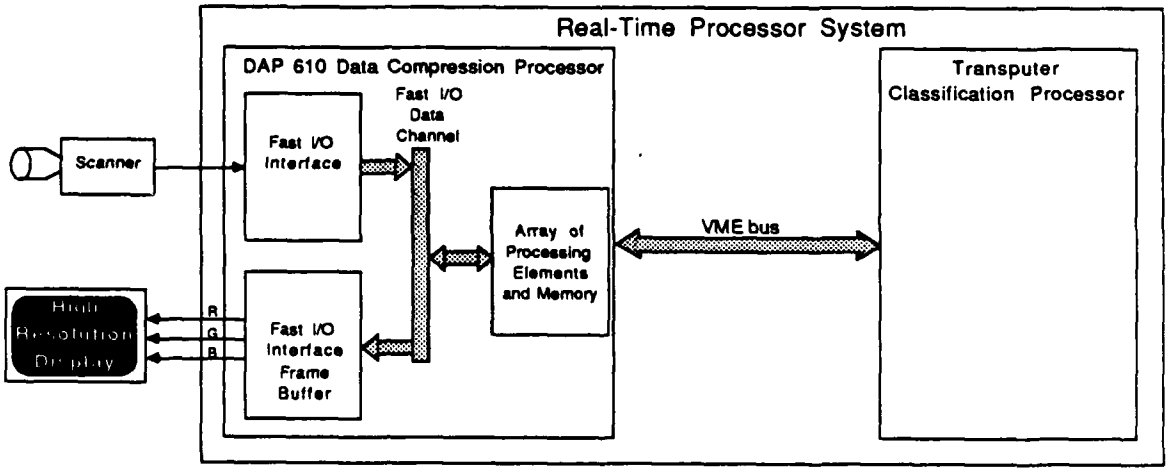

Figure 1. Real-Time Processor

1.1 Organization of the Report

This report is organized into five sections. Section 1 summarizes the DAP's required processing rate and evaluation results. Section 2 describes the DAP 610 architecture. Section 3 evaluates the fast I/O devices that allow fast input of sensor data to the DAP and output of display data from the DAP. Section 4 presents all required array processing operations with their corresponding performance results as well as an overview of the data compression algorithm. Section 5 analyzes the DAP's VMEbus interface capabilities in relation to output from the Compression Processor to the Classification Processor, and input of results from the Classification Processor back to the DAP.

1.2 Required DAP 610 Processing Rate

The DAP will receive one frame of multispectral image data every second from the sensor. One frame includes six channels each of size 750 words by 350 lines of 16-bit words. In addition to this data, a variable number of null words will be added to the end of each scan line because of scan mirror speed variations from one line to the next. The amount of null words transmitted per frame is expected to be fixed at a 5 percent increase in the total data. This will allow a synchronous data transfer rate of 3,307,500 bytes per frame. The DAP will be programmed to select three of the six images received from the sensor for processing; thus, the DAP must be capable of processing at least three images per second.

In order for the Compression Processor to meet the real-time processing requirements shown in Figure 2, the DAP must acquire the image, perform all array processing operations, and handle image display output all within one second. The architecture of the DAP can handle simultaneous execution of these three operations without a large reduction in processing performance. This concurrent processing capability of the DAP is illustrated in the timeline diagram (Figure 2); it allows a full second for the execution of each of the three tasks.

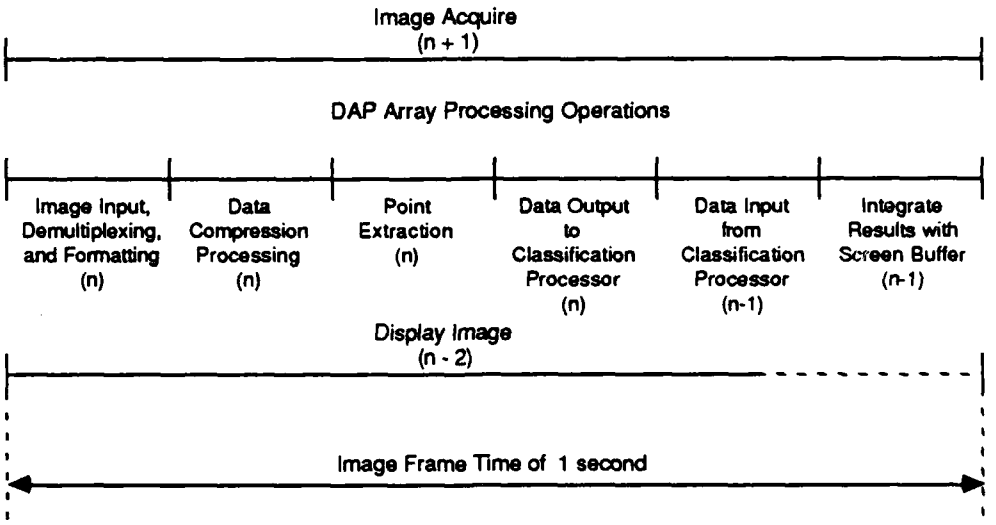

Figure 2. Data Compression Processing Timeline

1.3 Summary of Results

Evaluation results of the operations expected of the Compression Processor, as it is configured in the RTP, are summarized in this section. These operations will be presented in three categories: image acquisition, array processing, and image display. As illustrated in the timeline diagram of Figure 2, operations of these three categories are executed concurrently with each other; thus the operations of each category must be executed within the frame time of one second.

Image acquisition includes receiving one frame (6 channels) of image data from the sensor every second, identifying the start-of-line codes, and transferring three selected image channels to DAP memory for data compression processing. At this time, AMT's interface board, the FFIO, is available but not capable of identifying the start-of-line or selecting specific image channels for transfer to array memory. AMT's DIOC interface board could be used as an alternative since the DIOC is capable of data identification and manipulation operations. The DIOC is, however, still in the design phase at AMT and is not expected to be available until June 1989, assuming no design problems.

The second category (array processing) includes the operations that are executed using the processing power of the DAP array. The DAP array processing operations are shown in the timeline diagram of Figure 2, and are listed below with processing times.

	Activity	Time
1.	Move six images from D-Plane to DAP array memory	1.20ms
2.	Identify and format three images for DAP processing	not available
3.	Implement the data compression algorithm	100.5ms
4.	Organize results into coordinate and pixel value sets	not available
5.	Move data to and from the Classification Processor	48.0ms
6.	Merge classification results with display buffer	not available
7.	Output display buffer to graphics board for display	0.80ms

The above operations can be added together to determine the total expected array processing time. However, without essential information, ERIM cannot, at this time, determine if the DAP 610 is capable of performing all seven operations within the required processing time of one second.

High-resolution image display is the third category of operations executed by the Compression Processor. As listed in operation 6 above, AMT's high-resolution image display interface board depends on the processing power of the DAP to merge the classification results with the display image, and to handle the necessary data manipulation for paging or scrolling. Operations falling into this category only involve moving the display image from array memory to an external display system. There is ample time for AMT's display interface board to execute this operation within the frame time of one second.

In addition to the real-time hardware capabilities outlined above, software tools to create a real-time implementation program will be required. These tools (real-time support firmware) should address intertask communication, task scheduling, and attaching user written subroutines to interrupts. This software has not been identified as available from AMT for support of the DAP 610. The cost to develop and write a similar package could be prohibitive.

2.0 DAP 610 ARCHITECTURE OVERVIEW

The DAP 610 is a massively parallel computer system from Active Memory Technology (AMT^{TM}). The DAP, an acronym for Distributed Array of Processors, consists of four major functional units:

1. Array of Processor Elements with Local Memory
2. Master Control Unit and Code Memory
3. Host Connection Unit
4. Fast Data Channel

The interconnections between the major functional units of the DAP 610 are illustrated in Figure 3. The next four sections discuss each of the major functional units.

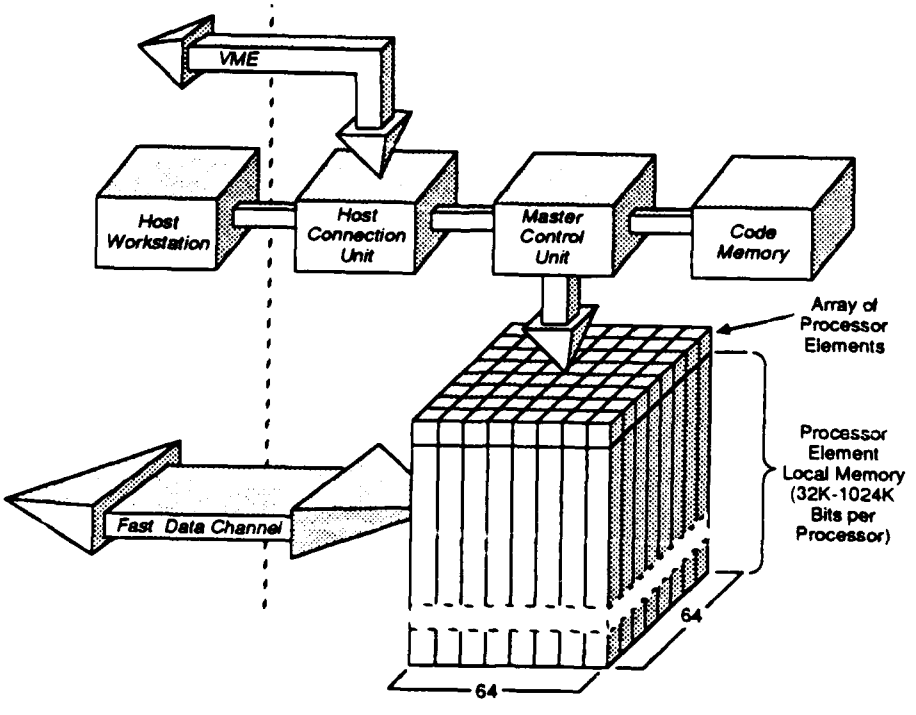

Figure 3. Major Functional Units of the DAP 610

2.1 Array of Processor Elements

The DAP 610 has 4096 Processing Elements (PEs) arranged in a 64 by 64, square matrix. The PEs are 1-bit processors with a standard 32 Kbits of local memory that is expandable to 1 Mbit. The PEs may be viewed as a horizontal array with the local memory extending downward from each PE. The PEs are connected via global row and column buses. In addition, each PE is connected to a north, south, east and west neighbor. Operands are often spread across the memory of a large number of PEs. All PEs selected as active simultaneously execute the same instruction, using the same bit address within their own memory. This type of implementation is referred to as a Single Instruction/Multiple Data (SIMD) stream architecture.

2.2 Master Control Unit and Code Memory

The DAP is controlled by the Master Control Unit (MCU), which fetches and interprets instructions from code memory. The MCU code memory has minimum and maximum sizes of 512 Kbytes and 2 Mbytes, respectively. The MCU controls the DAP 610 processor array by broadcasting instructions to the array of processing elements. The MCU also executes nonarray control and I/O instructions.

The array memory and the MCU control memory are mapped into the VMEbus address space. The MCU acts as a VME memory slave, sending data to the VMEbus only on request from the Host Connection Unit (HCU) or another VMEbus master. The MCU is not directly involved in moving data between array memory and an external VME device. The MCU execution of user application code is halted for three cycles to read array memory, and one cycle to write array memory. Read and write accesses must be performed as 32-bit transfers.

2.3 Host Connection Unit

The Host Connection Unit (HCU) links the DAP to a host. The HCU supports the Sun and VAX workstations as hosts under the standard Unix and VMS operating systems. The host is used for program development, debugging, loading, initiating and communicating

with DAP programs through the HCU. The HCU also performs VMEbus data movements, if needed, for a program executing on the DAP.

The HCU contains a Motorola 68020, 32-bit microprocessor for interface control. The HCU links to the MCU via the VMEbus and acts as a bus master. Permanent code can be stored in the HCU's 256 Kbytes of EPROM. For temporary data or code storage. 1 Mbyte of RAM is available on the HCU.

2.4 Fast Data Channel

A fast data link, shown in Figure 3 as the Fast Data Channel, is available for quick movement of data to and from array memory. A device coupler board moves data directly to and from the array memory via the DAP D-plane (an entire 64 by 64 array bit plane). A maximum transfer rate of 50 Mbytes per second is possible.

Transferring of data is orchestrated by a device coupler board. For data input to the DAP, the device coupler will first serially shift 128 successive 32-bit words from device coupler buffers to the D-plane. The device coupler will next issue a D-plane unload command with a destination address to the DAP. The DAP then takes one machine cycle of 100 ns to unload the D-plane. Shifting of data in or out of the D-plane by the device coupler takes place autonomously and in parallel with DAP array processing operations. DAP array processing operations are only suspended for one cycle when the D-plane is moved to or from array memory.

Up to four device coupler boards can be linked to the fast data channel. However, because there is only one D-plane, the device couplers must arbitrate for usage of the D-plane. Arbitration is implemented in parallel with the last bus cycle of the previous bus master. For connection to the fast data channel, a frame buffer to drive a high-resolution color display and general-purpose digital couplers are available.

3.0 PERFORMANCE ANALYSIS OF THE DAP FAST I/O DEVICES

The fast data link to array memory will be shared for image input and output. As indicated in section 2.4, an entire 64 by 64 bit plane buffer called the D-plane is used to move data between the I/O device couplers and array memory. Because there is only one D-plane, the device coupler boards must share accesses to this buffer. In the next two sections, the required input and output accesses to array memory are described. In section 3.1, AMT's FFIO board, which may be used to input multispectral sensor data to array memory via the D-plane, is evaluated. In section 3.2, AMT's graphics display board, which is used to output resulting image data to a high-resolution display, is evaluated.

3.1 Input Device Coupler Requirements and Evaluation Results

The direct interface of the sensor to the DAP is expected to be handled by AMT's FFIO (FIFO [First In First Out] Fast Input/Output) board. Although the FFIO is capable of accepting sensor data at the required 3,307,500 bytes per second, a communication protocol has not been developed for correct acceptance of sensor data. Once the sensor data is received, movement of sensor data into array memory is easily handled by the FFIO. However, it is not clear how the FFIO board will identify the start-of-line code for each scan line. Identification of the start-of-line codes will be required for proper storage of the images since a variable number of null words follows each scan line. If the FFIO is found to be inadequate, AMT's DIOC could be an alternative. The DIOC is capable of data identification and manipulation operations. However, the DIOC is still in the design phase at AMT and assuming no problems will not available until June 1989.

A certian amount of processing is required to reorganize the sensor data into a format suitable for DAP processing. This processing can happen either by the interface board (if it is capable) or by the DAP. One scan line of sensor data is presented in Table 1 to facilitate description of the required reorganization processing that follows. As shown, in Table 1, each scan line includes six channels of multiplex image data. From the six images, three must be selected and demultiplexed for storage in separate locations in array memory. In addition,

40 words of housekeeping data accompanies each scan line. This housekeeping data must be stripped from the actual image data. Lastly, 16-bit data is sent from the sensor, but 8-bit data will be used for data compression processing, most likely, the most significant 8 bits. AMT's DIOC board is expected to handle the required data manipulation processing. The FFIO is incapable of this image reformatting processing; thus, if the FFIO is used, the DAP will be burdened with an additional processing load (described in section 4.1).

Table 1. Format of the Data Received from the Sensor

Channel 1	Word 1
Channel 2	Word 1
Channel 3	Word 1
Channel 4	Word 1
Channel 5	Word 1
Channel 6	Word 1
Channel 1	Word 2
.	.
.	.
.	.
Channel 6	Word 750
Null	Words

Words 1-4	Start of Frame Code (SOF)
Words 5-32	Line Parametric Data
Word 33	REF 1
Words 34-743	Image Data
Word 744	REF 2
Words 745-750	End of Frame Code (EOF)

3.2 Output to the Frame Buffer for Image Display

High-resolution image displays are required of the DAP. For every frame of image data processed, three images are stored in array memory. One of these three images will be copied to the DAP screen buffer (a designated location within array memory). The image in the screen buffer will be overlaid with the results of the classification algorithm and then possibly be vertically decimated. Additional processing to format the display image in either the scrolling or paging modes will also be performed in the screen buffer.

Two image display methods (paging and scrolling) of the DAP are desired at this time. The exact method will be selected in the future after the appropriate human factors testing. To implement paging, the screen buffered image (1K by 1K) will be output to the graphics display card at a rate of one per six frames received if vertical decimation is used, and one per three frames if the screen buffered image is not vertically decimated. To implement scrolling, the display image is shifted up at the rate of 10 to 20 times per second with new image data added to the bottom of the display image to give the effect of a smooth scroll.

The processing required to execute scrolling, paging, and vertical decimation is implemented in the array of PEs under control of the MCU. The operations listed below are under the last section "Integrate Results with Screen Buffer" in Figure 2.

	Activity	Time
1.	Copy image to DAP screen buffer	not available
2.	Integrate classification results with screen buffer	not available
3.	Vertically decimate the screen buffer	not available
4.	Implement scrolling	not available
5.	Implement paging	not available

These operations are a small part of the required array processing operations implemented within the frame time of one second. The effect of these operations on the allowed one second of processing time has not been determined at this time.

Transfer of image display data stored in array memory to the high-resolution display takes place via the D-plane by the interface board. Moving data from the D-plane by the interface board takes place independently and has no effect on the array processing operations. With a frame time of one second, the MCU will only be suspended for 0.80 ms while image data is moved to the D-plane. The interface board will require access of the D-plane for the transfer of up to 1 Mbyte of image data per frame (one second). This can be accomplished because the D-plane will be unused for more than 90 percent of the given frame time.

4.0 DAP 610 PROCESSING SPEED PERFORMANCE ANALYSIS

In this section, a processing speed analysis of the DAP 610 is presented for performing the following operations:

1. Demultiplexing the multispectral image data
2. Separation of housekeeping data from image data
3. Conversion of 16-bit data to 8-bit data
4. Image formatting for image processing
5. Implemention of a data compression algorithm
6. Organization of data compression results into coordinate and pixel value sets

The timeline diagram of Figure 2 illustrates all the array processing operations that are required within the frame time of one second. The "Image Input, Demultiplexing, and Formatting" section includes the first four operations listed above. A description of the four operations is in the next section. Item 5 corresponds to the operation labeled "Data Compression Processing" in the timeline diagram. In section 4.2, the data compression algorithm (item 5) is introduced and presented with corresponding processing performance results for implementation on the DAP 610. Item 6 corresponds to the "Point Extraction" operation in the timeline diagram and is presented in section 4.3.

4.1 DAP Image Demultiplexing and Formatting

The format of image data received from the sensor is not suited to data compression processing. As indicated in section 3.1 and shown in Table 1, image data is received from the sensor in a six-channel line-multiplexed fashion. Demultiplexing of this image data into six separate images is required. The sensor data also includes housekeeping data that must be separated from the actual image data. As illustrated in Table 1 of section 3.1, 33 words of housekeeping data precede the image data and 7 words follow. In addition, 16-bit image values are received from the sensor but 8-bit values are required. Of the 16 bits only 12

are actually valid pixel bits. The most significant 8 bits of this 12 contain most of the pixel intensity information. Some type of (unspecified at this time) functional encoding will convert the 12-bit pixels into 8 bits for processing by the data compression algorithm. Lastly, the image data must be formatted into the crinkle format which is the format required by the DAP array for execution of the data compression algorithm.

The section of the timeline diagram of Figure 2 labeled "Image Input, Demultiplexing, and Formatting" includes implementation of the operations described above. The processing times of these operations are unavailable now except for the last operation.

	Activity	Time
1.	Demultiplexing of the three images	not available
2.	Separation of housekeeping data	not available
3.	Convert 16-bit words to 8 bits	not available
4.	Format the image into the crinkle format	5.0ms

These operations are a small part of the required DAP array processing operations implemented within the frame time of one second. The effect these operations will have on the allowed one second of processing time cannot be determined at this time.

4.2 DAP 610 Data Compression Processing

The approach taken to analyze the implementation of the data compression algorithm on the DAP 610 is described in this section. Section 4.2.1 presents an overview of the data compression algorithm. Section 4.2.2 gives a processing speed evaluation of the data compression algorithm implemented on the DAP.

This DAP 610 data compression processing analysis uses processing times calculated by AMT for their DAP 610. The processing times calculated were for 10 major suboperations used in the data compression algorithm, because the data compression algorithm can be described in terms of the ten identified suboperations. After describing the data compression algorithm in terms of the identified major suboperations, a total processing time is calculated using

the processing times measured by AMT. AMT was given the constraint of frame-by-frame processing on an image size of 700 pixels by 350 lines for determination of the processing times. In addition, the data precision required for the individual operations was fixed at the minimum amount of precision that the operation requires to avoid overflow. This data precision is different from the original C program written by WES (Waterways Experiment Station, U.S. Army Corps of Engineers).

4.2.1 Data Compression Algorithm Overview

For each frame of data received from the sensor, a set of three images are passed to array memory for data compression processing. The data compression operations reduce the image data to a small set of (x, y) coordinates and pixel values that identify potential objects of interest. For each (x, y) location, a pixel value at the specified location is returned from each of the three processed images. This resulting data is output to the Classification Processor.

The data compression algorithm presented in the following sections is a translation of a WES program written in the C programming language. In section 4.2.1.1, an operational flow diagram of the data compression algorithm is presented. Following in section 4.2.1.2, a more detailed description of the algorithm is given.

4.2.1.1 Flow Diagram Representation

The flow diagram representation (Figure 4) conveys much information about the algorithm. The operations that can be processed in parallel are shown horizontal to each other. Each of the three image channels (difference, sum, and thermal) can be processed in parallel as illustrated. Operations that are contingent upon the completion of a previous operation are shown with input arrows which represent the result of the previous operation. The resulting data precision for each of the major operations performed can be seen by the size of the resulting data buses going between blocks.

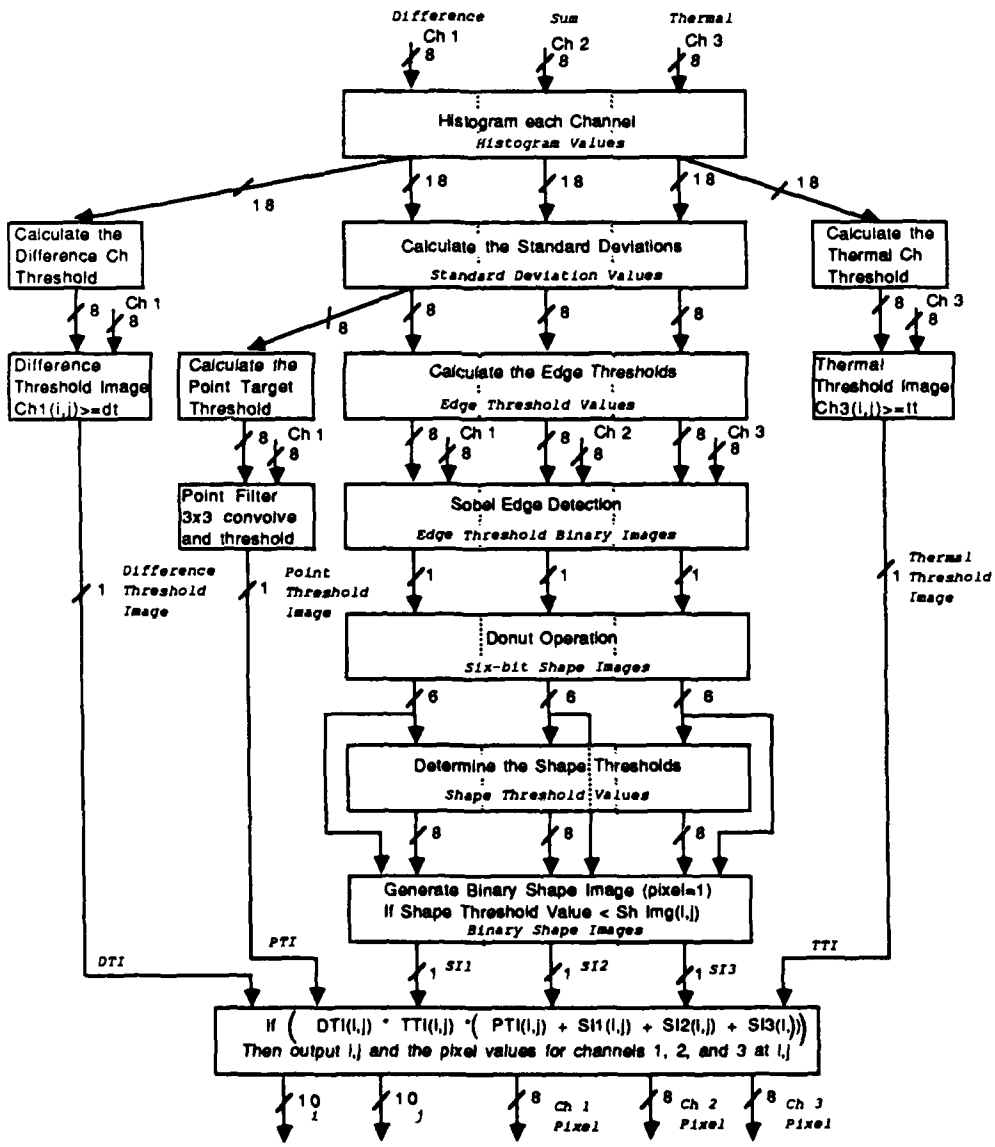

Figure 4. Data Compression Flow Diagram

4.2.1.2 Data Compression Image Processing Operations

The data compression algorithm begins by performing an edge detection operation (described in section 4.2.1.2.1) on the three input images: the difference, sum, and thermal images. The resulting edge detected binary images are processed for object detection and result in a corresponding binary shape image for each of the original three images. The object detection operations are described in more detail in section 4.2.1.2.2. Single point objects are detected in the difference image by the point target operations described in section 4.2.1.2.3. In addition, to exclude image information that is not expected to be a part of a potential object, the original thermal and difference images are thresholded. A description of this thermal and difference image thresholding is given in section 4.2.1.2.4.

The results of the above operations are used for the identification of image pixels in the original images that may be part of an object. The exact function that the results of the above operations play in this final detection step is described in section 4.2.1.2.5.

4.2.1.2.1 Edge Detection

The data compression algorithm begins by performing an edge detection operation on each of the three input images. Three binary images corresponding to the three image channels are generated. The generated images are flagged with a pixel of value one (depicting an edge) everywhere a pixel grey value greater than the calculated threshold value is detected.

Calculation of the image edge threshold values are implemented by using the results of a histogram operation from the corresponding image. The histogram data is used to calculate a corresponding mean value and then a standard deviation value. The calculated image edge threshold values are a function of the standard deviation values.

A Sobel edge detection operation (using a 3 by 3 kernel) is next performed on each of the three channels. The resulting images are then compared to their corresponding calculated image edge threshold values as described above. A new binary image is generated and contains a pixel of value one everywhere a pixel in the Sobel image is greater than the edge threshold value.

4.2.1.2.2 Object Detection

The detected edges flagged in the three binary images generated above are further processed to identify potential objects. A donut operation (7 by 7 convolution) is performed on each of the three binary images. The binary edge highlighted images are 6-bit images. The shape images are next compared to corresponding calculated shape threshold values. This thresholding separates background information from potential objects. Three new binary shape images with pixels of value one, indicating the presence of an object, are the end result of this object detection processing.

The three binary shape images (difference, sum, and thermal) generated will be used in a final thresholding operation (described in section 4.2.1.2.5) to identify potential objects for processing by the Classification Processor.

4.2.1.2.3 Point Target Detection

The point target detection operation is used to identify targets that register as a single pixel point in the difference image. If the difference between a center pixel and its neighbors is greater than 20 times the standard deviation (of this difference image), then a potential target that registered as a point has been detected. This point is flagged in a newly generated binary image as a pixel of value one.

The point target binary shape image generated here for the difference image will be used in a final thresholding operation (described in section 4.2.1.2.5) to identify potential objects for processing by the Classification Processor.

4.2.1.2.4 Difference and Thermal Image Thresholding

Most nonobject information is removed from the difference and thermal images by the thresholding operations described here. A binary image for each of the images processed is generated with pixels of value one everywhere potential objects may exist.

The difference and thermal images are each histogrammed and the resulting histogram data is used to calculate corresponding threshold values. Each threshold value is an 8-bit grey level value that represents a single pixel value. For the difference image, 90 percent

of the pixels in the image have a grey value below the identified threshold value. For the thermal image, 25 percent of the pixels in the image have a grey value below the identified threshold value.

The corresponding threshold values are then used to generate a corresponding binary image with pixels of value one everywhere an original image pixel value is greater than the corresponding threshold value.

The thermal and difference threshold binary images generated will be used in a final thresholding operation (described in section 4.2.1.2.5) to identify potential objects for processing by the Classification Processor.

4.2.1.2.5 Objects Flagged for Processing by the Classifier

The operations above result in the generation of six binary images. These binary images are analyzed as a final data compression operation to identify potential objects for processing by the Classification Processor.

If a pixel at the same location in the thermal and difference threshold binary images are pixels of value one, and if any one of the binary images listed below also has a pixel of value one, then the (x, y) coordinates for the given location and pixel values from the original three images at the given location are sent to the Classification Processor.

- Point target binary image
- Shape binary image corresponding to the difference channel
- Shape binary image corresponding to the thermal channel
- Shape binary image corresponding to the sum channel

4.2.2 DAP 610 Data Compression Processing Analysis Results

The processing speed performance analysis of the DAP 610 for execution of the data compression algorithm is presented in this section. First, in section 4.2.2.1, a list of 10 major data compression suboperations that were used to evaluate the capabilities of the DAP 610 are given. In section 4.2.2.2, the processing times of the listed major suboperations are

given. Finally, in section 4.2.2.3, the data compression algorithm is described in terms of the major suboperations and a total DAP 610 data compression implementation processing time is given.

4.2.2.1 List of the Data Compression Operations Evaluated

The data compression algorithm can be described by the 10 operations listed below.

1. Perform a histogram on the image
2. Perform Summation and Comparison

$$\sum_{i=0}^{255} h[i]\ddagger \text{ until } \geq \text{18-bit test Value}$$

3. Generate a binary image base on image thresholding using an 8-bit constant as the threshold value.
4. Determine Average Pixel value

$$\sum_{i=0}^{255} \left(\frac{h[i]}{700 \times 350} \right) \times i$$

5. Determine Standard Deviation

$$\sqrt{\frac{\sum_{i=0}^{255} (i - Avpix)^2 \times h[i]}{700 \times 350}}$$

6. Perform a 3 by 3 Convolution; the convolution mask is 1 everywhere except the center which is 8.
7. Perform a 3 by 3 Sobel
8. Perform a Donut Operation on a 700 pixel by 350 line binary image; using a 7 by 7 kernel this operation generates a new 6-bit center pixel. The kernel mask is 1

‡ $h[i]$ are 18-bit values

everywhere except the center 3 by 3 neighborhood which is 0. The center pixel is the sum of pixels in the 7 by 7 kernel where the mask is 1.

9. Perform a pixel-by-pixel bit-wise logical OR operation on four 700 pixel by 350 line binary images.

10. Perform a pixel-by-pixel bit-wise logical AND operation on three 700 pixel by 350 line binary images.

4.2.2.2 Processing Times for the Operations Evaluated

The processing times for the 10 data compression algorithm suboperations, listed in the previous section, are given in this section. The DAP 610 processing times (Table 2) were calculated by AMT and were performed on images of size 700 pixel by 350 line image with 8-bit pixels.

Table 2. Process Times for 10 Data Compression Suboperations

	Operation	Process Time
1.	Perform Histogram	10.00ms
2.	Perform Summation and Comparison	0.10ms
3.	Generation of Binary Image based on Thresholding	0.13ms
4.	Determine Average Pixel Value	0.36ms
5.	Determine Standard Deviation	0.72ms
6.	Perform 3 by 3 Convolution	1.10ms
7.	Perform 3 by 3 Sobel	1.50ms
8.	Perform Donut Operation	10.00ms
9.	Perform Pixel by Pixel bit-wise logical OR	0.02ms
10.	Perform Pixel by Pixel bit-wise logical AND	0.03ms

4.2.2.3 Overall Data Compression Performance Results

The flow diagram in Figure 4 of section 4.2.1.1 illustrates that the data compression algorithm could be processed in parallel provided the processing hardware was available. The

architecture of the DAP 610 is not suited for a parallel implementation of the algorithm; thus, the performance analysis of the DAP 610 presented here is based on a serial implementation.

A serial implementation of the data compression algorithm can be described in terms of the 10 operations listed in section 4.2.2.1. The number of possible serial implementations of the data compression algorithm is unlimited. For this analysis the serial implementation selected was roughly obtained by serially walking through the flow diagram of Figure 4.

Starting at the top box of Figure 4 and moving down the main path implementing edge detection and then the object detection operations results in three binary shape images. Next, the path on the far left was implemented generating the difference threshold image. This is followed by implementation of the path on the far right that gives the thermal threshold binary image. Next, the second path from the left is executed to obtain a point target binary image. Finally, the last box in Figure 4 is implemented resulting in a binary image.

Pixels of value one in this final binary image indicate the presence of potential objects. The Classification Processor is sent a set of data for each pixel of value one. Each set is composed of an (x, y) coordinate (indicating the location of the pixel of value one) and three pixel values. A pixel, at the specified coordinate, is taken from each of the original three images.

In Table 3 on the next page, the operations of the selected serial implementation are listed with their corresponding processing times.

Table 3. Overall DAP 610 Data Compression Processing Time

Operation	Process Time
Edge Detection	
Histogram the Three Images	30.00ms
Determine Average Pixel Value for Each Input Image	1.08ms
Determine Standard Deviation for Each Input Image	2.16ms
Perform a 3 by 3 Sobel Operation on Each Input Image	4.50ms
Generate a Binary Edge Image for Each Input Image	0.30ms
Object Detection	
Perform a Donut Operation on Each Input Image	30.00ms
Histogram the Three Shape Images	30.00ms
Determine Corresponding Image Thresholds (Summation and Comparison)	0.30ms
Generate a Binary Shape Image for Each Input Image	0.30ms
Generation of the Difference Threshold Binary Image	
Determine Difference Image Threshold (Summation and Comparison)	0.10ms
Generate Difference Threshold Binary Image	0.13ms
Generation of the Thermal Threshold Binary Image	
Determine Thermal Image Threshold (Summation and Comparison)	0.10ms
Generate Thermal Threshold Binary Image	0.13ms
Point Target Detection	
Perform a 3 by 3 Convolution on the Difference Image	1.10ms
Generate a Point Target Binary Image	0.13ms
Final Thresholding Operation	
Perform Four Pixel by Pixel bit-wise Logical ORs	0.08ms
Perform Three Pixel by Pixel bit-wise Logical ANDs	0.09ms

The required DAP 610 processing time is 100.5 ms, assuming no dead time between the execution of the above operations and no image reformatting.

4.3 DAP organization of Data Compression Results

The data compression operations described in section 4.2 result in the generation of a binary image with pixels of value one indicating potential objects that are to be processed by the Classification Processor. For each pixel of value one a data set is output to the Classification Processor. A maximum of 5,000 data sets are expected. Each set of the 5,000 is composed of an (x, y) coordinate and three pixel values. A pixel, at the specified coordinate, is taken from each of the original three images.

The DAP array under the control of the MCU will be used to identify the pixels of value one, determine their coordinates, and then obtain the pixel values from the original three images at the corresponding location. The "Point Extraction" section of the timeline diagram of Figure 2 encompasses the operations described above. The processing effect these operations will have on the one second frame time has not yet been determined.

5.0 PERFORMANCE ANALYSIS OF THE DAP VMEBUS INTERFACE

Data movement between the two main components of the RTP (the DAP 610 and the Classification Processor) is an additional source of processing load. Communication in both directions between the two processors is carried out over a VMEbus link. A compressed set of data resulting from the data compression processing is first output to the Classification Processor. The Classification Processor then categorizes and groups the pixel data received from the DAP into individual objects. The objects identified by the Classification Processor are given a rank value corresponding to the likelihood that they belong to an object of interest and are then returned to the DAP for display.

The processing load to implement the described data transfers total 3.2 ms of MCU time and 48.0 ms of VMEbus usage. The calculation of these processing times are presented in the next section.

5.1 DAP VMEbus Communication Processing Load

The data compression operations result in a maximum of 5,000 sets of coordinates and pixel values indicating potential objects of interest. This data set is output to the Classification Processor. Each set of the 5,000 is composed of an (x, y) coordinate and three pixel values. A pixel, at the specified coordinate, is taken from each of the original three images. Each set of the 5,000 can be expressed by two 32-bit words where the (x, y) coordinates (16 bits each) are one 32-bit value and the three 8-bit pixel values are another 32-bit value. The 5,000 sets of data amount to a total of 10,000 32-bit words or 40,000 bytes.

The classification results sent back for display require the transfer of a maximum of 1,000 objects. Each object of the 1,000 is composed of an (x, y) coordinate and one pixel rank value. Each set can be expressed by two 32-bit words where the (x, y) coordinates (16 bits each) are one 32-bit value and the rank (8-bit pixel) value as the other 32-bit value. The 1,000 objects of data amount to a total of 2,000 32-bit words or 8,000 bytes.

As indicated in section 2.2, the addressing of array memory from the VMEbus halts the DAP Master Control Unit (MCU). The MCU is stopped for 3 cycles to read and 1 cycle to write array memory. Read and write accesses are performed as 32-bit transfers. A total of 10,000 read and 2,000 write accesses are expected as presented above. This requires the MCU to be halted for a total of 32,000 bus cycles. With a bus cycle of 100 ns, the total MCU idle time is 3.2 ms.

The MCU is not directly involved in moving data between array memory and an external VME device. The MCU is only halted from continuing the execution of user application code for the 3.2 ms indicated above. Moving 32 bits of data between array memory and an external VME device is specified by AMT to take as much as $1\mu s$. This rate can only be maintained if a VMEbus master device could keep up. A more conservative figure of $4\mu s$ per 32-bit transfer will be used in the calculations that follow.

The total amount of data expected to be transferred over the VMEbus is 12,000 32-bit words. At a rate of $4\mu s$ per 32-bit transfer, this amounts to a total data transfer time of 48 ms. The effect of this processing load on the one second of frame time is only 0.48 percent. This processing load is for the data output and data input operations labeled in the timeline diagram of Figure 2.